User Manual

GoPro Trails

Images and Graphics, Project Course CDIO

TSBB11

Group Members

Daniel Cranston, dancr948

Carl Ekman, carek025

Lisa Eriksson, liser858

Freja Fagerblom, frefa105

Filip Skarfelt, filsk543

Version 1.0

December 17, 2018

Contents

1 Introduction

This document explains how to use the GoPro Trails software. The software produces a trajectory and a 3D model of a video sequence supplied by the user. The trajectory is visualized in an interactive map service and the 3D model is shown in MeshLab.

2 Installation

Follow the steps below to install the software. Observe that the program only works on Linux.

1. Make sure Ceres is installed. It can be installed using your system package manager. For instance, using Ubuntu you can install Ceres by executing

   ```
   $ sudo apt-get install libceres-dev
   ```

2. Clone the Git repository available at `https://gitlab.ida.liu.se/cdio-gopro/gopro`

   ```
   $ git clone https://gitlab.ida.liu.se/cdio-gopro/gopro
   ```

3. Make sure `pip` is installed and updated (at least version 18.1). Update `pip` by executing the following command

   ```
   $ pip install --upgrade pip
   ```

4. Install GoPro Trails by executing the following

   ```
   $ pip install ./path/to/cloned/gopro/directory
   ```

5. Verify that the program was installed correctly by executing

   ```
   $ gopro-trails --help
   ```

6. To use the surface reconstruction and 3D visualization features, MeshLab needs to be installed and available on the system `PATH`. MeshLab can be installed by downloading the Linux snap from `http://www.meshlab.net/#download`. Observe that at least version 2016.12-2 should be used.

2.1 Developer Installation

When developing the software further the flag `-e` can be used in step 4 in the previous section. This flag enables continuous update of the program while editing is performed. Hence, the developer does not have to uninstall and install the program after each change in the code. Step 4 can therefore be changed to the following

```
$ pip install -e ./path/to/cloned/gopro/directory
```

3 User Instructions

To print an explanation of the command line interface, execute the following command

```
$ gopro-trails --help
```

This prints the following

```
usage: gopro-trails [-h] {run,map,mesh,create-config} ...

Performs structure from motion and georeferencing.

optional arguments:
```

```
  -h, --help            show this help message and exit

Subcommands:
  {run,map,mesh,create-config}
                        Available subcommands.
    run                 Runs the SFM pipeline and produces output files.
    map                 Opens default web browser and displays a map
                        with GPS coordinates and SFM trajectory.
    mesh                Opens Meshlab and displays surface
                        reconstruction of generated 3D points.
                        Surface reconstruction will be
                        performed if not already found in the
                        output directory.
    create-config       Creates a configuration file in current
                        directory or specified directory with
                        tunable parameters.
```

As explained in the help message, the program is executed by supplying one of the sub-commands. To learn more about a specific sub-command execute `gopro-trails <sub-command> --help`. The sub-commands are further explained in the following sections.

3.1 Sub-command Run

This command executes the main part of the program. It will generate tracks (unless pre-computed), perform structure from motion, georeferencing of trajectory and store all output files in the output directory.

To print an explanation of the sub-command `run`, execute the following command

```
$ gopro-trails run --help
```

This prints the following

```
usage: gopro-trails run [-h] [-c CONFIG] [-t TRACKS] [-dd]
                        [-o OUTPUT_DIR] [-cp PARAM_PATH]
                        video-path

Runs the SFM pipeline and produces output files. Generated output files
will be stored in a directory called gopro-trails-output created in the
current directory. NOTE: previous files in the output directory will be
overwritten if the same directory is used again.

positional arguments:
  video-path            Path of the GoPro video file to use.

optional arguments:
  -h, --help            show this help message and exit
  -c CONFIG, --config CONFIG
                        Path to config file. If not given, default path
                        will be used.
  -t TRACKS, --tracks TRACKS
                        Path to hdf file containing pre-computed tracks.
                        If not given, tracks will be generated.
  -dd, --disable-dense  Disable computation of dense correspondences
                        and dense 3D points.
  -o OUTPUT_DIR, --output-dir OUTPUT_DIR
                        Specify path to output directory (will be
                        created if it does not exist). If not
                        specified, output directory will be
```

```
                        created in the current directory
-cp PARAM_PATH, --camera-parameters PARAM_PATH
                        Optional path to an hdf file with custom
                        calibration parameters or one of the built
                        in camera parameter files by specifying
                        "1080p30fps" or "1080p60fps".
                        (Default value: "1080p30fps")
```

3.1.1 Positional Arguments

This sub-command takes one positional argument, supplied after all the optional arguments.

video-path

Path of the GoPro video file to use.

3.1.2 Optional Arguments

This sub-command supports several optional arguments.

--help

The `--help` argument displays the the description for the sub-command and all possible arguments with belonging descriptions. The print after running the program with this argument is displayed above in Section 3.1.

--tracks

Path to an HDF file containing pre-computed tracks. When not given the tracks are generated for the video.

--disable-dense

If this switch is given, the computation of dense correspondences and 3D points will be skipped and the more sparse 3D points from the SFM system will be output.

--output-dir

Specifies an output directory that will be created and hold all the output files. If not given, a folder named `gopro-trails-output` will be created in the current directory. Observe that previous files in the output directory will be overwritten if the same directory is used again.

--camera-parameters

Used to specify a custom HDF file with camera parameters or one of the built in camera parameter files (”1080p30fps” or ”1080p60fps”). The custom file should have the following fields:

- **size**: Video size as [width, height].
- **readout**: Readout time in seconds.
- **K**: Intrinsic camera parameters, 3x3 matrix.
- **wc**: Distortion center ω_c, 2-element vector.
- **lgamma**: Distortion parameter γ, scalar.
- **fps**: Frames per second.

3.2 Sub-command Map

This command opens the default web-browser and displays the created SFM trajectory together with the GPS coordinates in a map service.

To print an explanation of the sub-command `map`, execute the following command

```
$ gopro-trails map --help
```

This prints the following

```
usage: gopro-trails map [-h] [gopro-trails-output]

Opens default web browser and displays a map with
GPS coordinates and SFM trajectory.

positional arguments:
  gopro-trails-output  Path to GoPro-trails output directory.
                       (Default value: ./gopro-trails-output)

optional arguments:
  -h, --help           show this help message and exit
```

3.2.1 Positional Arguments

Sub-command `map` only has one positional argument. It is described below.

gopro-trails-output

Path to GoPro-trails output directory. This only needs to be specified if the user has chosen an own output directory when running the program. If this argument is not specified then the default directory `./gopro-trails-output`, where the output files automatically are stored, will be used.

3.2.2 Optional Arguments

Sub-command `map` only has one optional argument. It is described below.

--help

The `--help` argument displays the the description for the sub-command and all possible arguments with belonging descriptions. The print after running the program with this argument is displayed above in Section 3.2.

3.3 Sub-command Mesh

This command performs surface reconstruction from the generated 3D points and opens Meshlab to display the results. If this command already has been executed, then the previous surface reconstruction will be used and opened in Meshlab.

To print an explanation of the sub-command `mesh`, execute the following command

```
$ gopro-trails mesh --help
```

This prints the following

```
usage: gopro-trails mesh [-h] [-dds] [-cds] [gopro-trails-output]

Opens Meshlab and displays surface reconstruction of
generated 3D points. Surface reconstruction will be
performed if not already found in the output directory.

positional arguments:
  gopro-trails-output   Path to GoPro-trails output directory.
```

```
                        (Default value: ./gopro-trails-output)

optional arguments:
  -h, --help            show this help message and exit
  -dds, --dot-decimal-sep
                        Use dot instead of system separator as floating
                        point decimal separator in ply file for meshlab
                        (Try this if file fails to load in meshlab)
  -cds, --com-decimal-sep
                        Use comma instead of system separator as
                        floating point decimal separator in ply
                        file for meshlab
                        (Try this if file fails to load in meshlab)
```

3.3.1 Positional Arguments

Sub-command `mesh` only has one positional argument. It is described below.

gopro-trails-output

Path to GoPro-trails output directory. This only needs to be specified if the user has chosen an own output directory when running the program. If this argument is not specified then the default directory `./gopro-trails-output`, where the output files automatically are stored, will be used.

3.3.2 Optional Arguments

Sub-command `mesh` has two optional arguments. These are described below.

--help

The `--help` argument displays the the description for the sub-command and all possible arguments with belonging descriptions. The print after running the program with this argument is displayed above in Section 3.4.

--dot-decimal-sep

The `--dot-decimal-sep` flag forces dots to be used in the ply file copy that is read by Meshlab, regardless of system separator. **Note:** if Meshlab is unable to open the ply file with the default setting (system separator), then the problem might be solved by changing the decimal separator. The original ply file from the `run` command is never modified in-place.

--com-decimal-sep

The `--com-decimal-sep` flag forces commas to be used in the ply file copy that is read by Meshlab, regardless of system separator. See notes for `-dds`.

3.4 Sub-command create-config

This command creates a GoPro Trails configuration file in the current directory, or the directory specified. The configuration file contains tunable parameters that affect the results of the program.

To print an explanation of the sub-command `create-config`, execute the following command

```
$ gopro-trails create-config --help
```

This prints the following

```
usage: gopro-trails create-config [-h] [config-file]

Creates a configuration file in current directory or specified
```

```
directory with tunable parameters.

positional arguments:
  config-file  Configuration file destination path.
               (Default value: ./gopro-trails.cfg)

optional arguments:
  -h, --help   show this help message and exit
```

3.4.1 Positional Arguments

Sub-command `create-config` only has one positional argument. It is described below.

config-file

Path to configuration file that will be created. If not specified it will be called `gopro-trails.cfg` and be placed in the current directory.

3.4.2 Optional Arguments

Sub-command `create-config` has one optional argument. It is described below.

--help

The `--help` argument displays the the description for the sub-command and all possible arguments with belonging descriptions. The print after running the program with this argument is displayed above in Section 3.4.

4 Configuration File

The configuration file created via the sub-command `create-config` (see Section 3.4) contains different groups of parameters. The created configuration file parameters will be filled with default values. The default values are not considered optimal in any way and should most likely be tuned to get good performance. The groups and their parameters are described in the subsections below.

4.1 Kontiki Parameters

The group called *KontikiParams* contains parameters which effects the optimization process run by Kontiki. These parameters are described below.

huber_c1

Adjusts the impact of outliers for the first optimization phase (there are 4 phases in total). The larger value, the larger impact of outliers.

huber_c2

Adjusts the impact of outliers for the second optimization phase (there are 4 phases in total). The larger value, the larger impact of outliers.

huber_c3

Adjusts the impact of outliers for the third optimization phase (there are 4 phases in total). The larger value, the larger impact of outliers.

huber_c4

Adjusts the impact of outliers for the fourth optimization phase (there are 4 phases in total). The larger value, the larger impact of outliers.

max_iter1

Sets the maximum number of iterations during the first optimization phase.

max_iter2

Sets the maximum number of iterations during the second optimization phase.

max_iter3

Sets the maximum number of iterations during the third optimization phase.

max_iter4

Sets the maximum number of iterations during the fourth optimization phase.

initial_inv_depth

Specifies the initial inverse depth of all landmarks. This is included as a debug tool and should generally not be changed from the default value.

acc_bias

Specifies the accelerometer bias. Observe that this value is only used if the parameter `zero_bias` is set to `False`.

zero_bias

Specifies if no bias should be used during optimization. If set to `False`, accelerometer bias will be considered during optimization.

rand_reference

Specifies if a random observation should be the reference to a landmark. If set to `False`, then the first observation is used as reference.

gyro_std

Specifies the standard deviation of the gyroscope noise, which is used to estimate a better weighting.

acc_std

Specifies the standard deviation of the accelerometer noise, which is used to estimate a better weighting.

max_error_first_cull

Specifies the threshold for outlier detection in pixels. This threshold is used in the first outlier elimination. If the mean of the residuals for a landmark is above the threshold, the landmark is culled.

max_error_second_cull

Specifies the threshold for outlier detection in pixels. This threshold is used in the first outlier elimination. If the mean of the residuals for a landmark is above the threshold, the landmark is culled.

q_gyro

Specifies a quality measure for how well you want the SO3-spline to model the measured data. This affects the knot spacing of the spline.

q_acc

Specifies a quality measure for how well you want the R3-spline to model the measured data. This affects the knot spacing of the spline.

num_obs_per_frame

Specifies how many observations per keyframe are to be used. The selected observations will be evenly spaced throughout the image.

keyframe_ratio

Specifies the threshold of common tracks between keyframes. A higher value will result in more keyframes.

keyframe_min_dist

Specifies the minimum spacing between keyframes.

precalc_rel_pos

A debug parameter that specifies whether the relative pose between the IMU and the camera should be preprocessed instead of letting Kontiki handle the transformation. Shouldn't affect the result.

plot_progress

Specifies whether plots of the residuals and trajectory together with 3D points should be plotted during the optimization process. This is best used as a debug tool, or to try out new parameters, since it is easy to abort early, but shouldn't be used in the general case since it affects the optimization procedure.

plot_phase

Specifies whether or not to plot the residuals and trajectory with 3D points after optimization phase.

plot_final

Specifies whether or not to plot the end result of the residuals and trajectory with 3D points after the whole Kontiki pipeline. This halts the program until the plots are closed, in order to give the user time to view the plots before the program exits after finishing.

4.2 Dense Correspondences Parameters

The group called *DenseCorrespondencesParams* contains parameters regarding the densification of the point cloud. These parameters are described below.

baseline_dist

Specifies the minimum distance in meters between image pairs used in PatchMatch

keyframe_stride

Specifies how many frames to skip before choosing the next image pair

scale_factor

Specifies how much the images should be downscaled before PatchMatch is executed. A value of 2 results in a down-scaling of half the original size.

kernel_size

Specifies the patch size to be used in PatchMatch.

pm_iter

Specifies how many iterations PatchMatch will run.

max_reproj_error

Specifies the maximum reprojection error (in pixels) tolerated when removing outlier landmarks generated from PatchMatch correspondences. Landmarks with a larger error than this value will be removed.

visualize

Boolean specifying whether or not to visualize the correspondences found by PatchMatch.

4.3 IMU Parameters

The group called *ImuParams* contains parameters regarding the IMU. These parameters are described below.

true_rate

Specifies the sampling rate of the IMU.

offset

Specifies the time offset between when the camera and the IMU starts recording.

4.4 Georeference Parameters

The group called *GeoreferenceParams* contains parameters which effects the georeferencing step. These parameters are described below.

gps_outlier_threshold

Threshold in metres for deviation from short time median GPS point.

gps_outlier_median_interval

Time interval used to compute short time median of GPS points.

trajectory_sample_freq

Sampling frequency (sample/s) used to sample the SFM trajectory.

4.5 Tracking Parameters

The group called *TrackingParams* contains parameters which effects the tracking step. These parameters are described below.

min_track_length

Specifies the minimum amount of frames in which a track has to be alive in order to be saved.

backtrack_length

Specifies the amount of frames in which to backtrack.

min_points

Specifies how many points to track in a frame.

min_distance

Specifies the minimum distance between existing and new tracks when finding new tracks.

win_size

Specifies the patch size of the tracker.

visualize

Specifies whether or not to visualize the tracking procedure.

4.6 Point Cloud Parameters

The group called *PointcloudParams* contains parameters which effects the surface reconstruction step. These parameters are described below.

voxel_size

Sets the voxel size for down-sampling point cloud during processing stage to create a uniformly down-sampled point cloud from a regular voxel grid.

nb_neighbors

This specifies how many neighbors are taken into account in order to calculate the average distance for a given point in the outlier removal stage.

std$_ratio$

This parameter sets the threshold level based on the standard deviation of the average distances across the point cloud. A lower threshold will remove more points.

radius

Specifies search radius for normal estimation. Neighboring points within the radius will affect the normal.

max_nn

This sets the maximum nearest neighbors that are taken into account for normal estimation to save computation time.

min_dist

The minimum distance from reference observation position a landmark needs to have to be extracted from the Kontiki Object to the point cloud.

max_dist

The maximum distance from reference observation position a landmark needs to have to be extracted from the Kontiki Object to the point cloud.